Calculs d'états globaux remarquables dans un système réparti

Jean-Michel Hélary, Noël Plouzeau, Michel Raynal

INRIA

UNITÉ DE RECHERCHE
INRIA-RENNES

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France

Tél.:(1) 39 63 55 11

Rapports de Recherche

N° 816

CALCULS D'ETATS GLOBAUX REMARQUABLES DANS UN SYSTEME REPARTI

Jean-Michel HELARY
Noël PLOUZEAU
Michel RAYNAL

AVRIL 1988

RR 816

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE
ET SYSTÈMES ALÉATOIRES

Campus Universitaire de Beaulieu
35042 - RENNES CÉDEX
FRANCE
Téléphone : 99 36 20 00
Télex : UNIRISA 950 473 F
Télécopie : 99 38 38 32

PUBLICATION INTERNE N° 396

MARS 1988

22 PAGES

Calculs d'états globaux remarquables dans un système réparti

Computing particular snapshots in distributed systems

J.-M. Hélary N. Plouzeau M. Raynal*

*I.R.I.S.A. Campus de Beaulieu 35042 RENNES CEDEX, E-mail: raynal@irisa.fr

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE
(L. A. 227)
UNIVERSITÉ DE RENNES 1 I. N. S. A. DE RENNES

INSTITUT NATIONAL DE RECHERCHE
EN INFORMATIQUE ET EN AUTOMATIQUE
(LABORATOIRE DE RENNES)

Résumé

Le calcul d'états globaux cohérents est un problème fondamental dans le contexte des applications et des systèmes répartis. Un état global est caractérisé par l'état de chacun des sites et l'état de chacun des canaux (les messages en transit) à l'instant considéré. Parmi tous les états dans lesquels passe un tel système, certains peuvent être caractérisés par un prédicat de stabilité P : si P est vérifié sur un état S il est alors vérifié sur tous les états accessibles depuis S. Plusieurs algorithmes ont été proposés pour détecter de telles propriétés. On s'intéresse dans cet article à une classe particulière d'états globaux qui ne peut être caractérisée par une propriété de stabilité : ceux dans lesquels il n'y a pas de messages en transit. Après avoir caractérisé ces états et montré leurs intérêts, les principes algorithmiques sur lesquels repose leur détermination sont exposés ; celle-ci nécessite la visite des sites du système. En fonction de la technique de parcours choisie pour réaliser cette visite, une famille d'algorithmes est envisageable. Parmi ceux-ci l'algorithme s'appuyant sur une visite à l'aide d'un anneau est présenté et prouvé. La présentation des principes et de l'algorithme proposé se veut analytique ; les outils algorithmiques utilisés sont introduits (marqueurs, comptage, parcours) pour rendre compte des propriétés caractérisant les états globaux remarquables recherchés. (An english translation of this paper is available from the authors.)

Abstract

Computing consistent snapshots is a major problem faced at in distributed systems and computations. A snapshot is made of the states of the computing nodes and of the network channels (*i.e.* messages in transit taken at a given time. A subset of the states that a computation may reach can be defined by a stability predicate P. If P is true for some state S then it is also true for every state reachable from$S$. Several algorithms detecting the occurence of stability properties are known. In this paper we focus at a particular class of instable global states : the states with no message in transit. A caracterisation of these states and their interest are given, then the principles of their detection are exposed. As detecting these states requires a network traversal, choosing a particular implementation of this traversal gives one instance from a family of algorithms. The algorithm based on a virtual hamiltonian cycle (a ring traversal) is presented and proved to be correct. The exposition of the principles and of the ring algorithm is made in an analytic way: algorithmic techniques are exposed separately (markers, message counting, network traversal) to lay down the snapshot definition properties.

1 Le calcul d'états globaux

1.1 Quelques intérêts des calculs d'état globaux

Le calcul d'un état global relatif à une application ou à un système réparti est un problème fondamental de l'algorithmique distribuée, tant par son intérêt à offrir des solutions à certains problèmes de contrôle que par les mécanismes fondamentaux de calcul réparti qu'exhibent les solutions qui sont proposées pour le résoudre.

L'obtention d'un état global est en effet intéressante à plus d'un titre. Il y a tout d'abord la détection de propriétés stables sur l'état du système. Une propriété de stabilité est une propriété qui, une fois vérifiée, le reste ; c'est notamment le cas de la terminaison et de l'interblocage [2,14,3,5,7]. Un site peut alors calculer un état global, puis calculer la propriété sur cet état ; si celle-ci est vérifiée, tous les états dans lesquels passera le système la vérifieront aussi ; dans le cas contraire le site peut calculer un nouvel état global et recommencer.

Un second intérêt réside dans la généralisation de cette technique : elle permet d'adapter des algorithmes de contrôle centralisés au contexte réparti, en les appliquant à des états globaux du système réparti. Un autre intérêt réside dans la résistance aux pannes, pour un système réparti. Si l'on connaît un état global du système il est possible de rétablir le système dans cet état après une panne [11] ; un état global définit en effet un point de reprise à partir duquel l'exécution peut être relancée.

1.2 Problématique du calcul de l'état global

Le calcul d'un état global est un problème difficile à résoudre. Le système réparti, sur lequel on veut calculer un tel état, est formé de n sites : $X = \{P_1, \ldots, P_n\}$, connectés par des canaux de communication unidirectionnels ne déséquençant pas les messages (l'ensemble des arcs du graphe

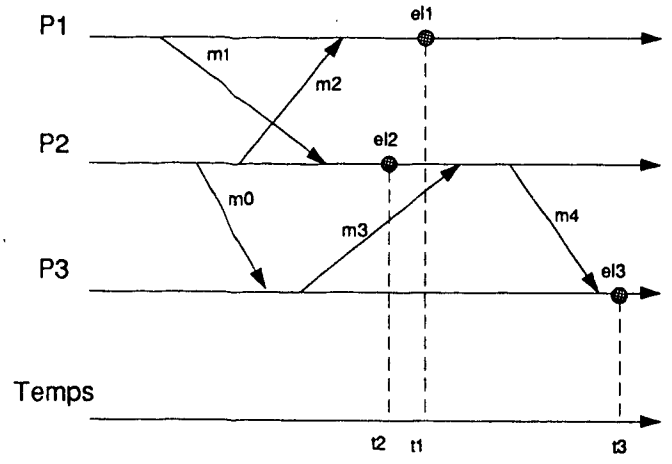

Figure 1 : Un exemple de prise d'état global

de communication est noté Γ). Les temps de transfert des messages sont supposés arbitraires mais finis. Le graphe orienté $G = (X, \Gamma)$ formé par les sites et les canaux est supposé fortement connexe. Considérons à titre d'exemple un système formé de 3 sites dont une exécution est représentée à la figure 1 ; les traits horizontaux modélisent pour chaque site l'évolution du temps, les flèches représentent les échanges de messages (la réception d'un message étant toujours postérieure à son émission, les flèches sont orientées de la gauche vers la droite : un transfert de message prend un certain temps).

A l'instant t_i, le site P_i décide de mémoriser son état local $el(t_i)$ (que nous noterons el_i pour simplifier l'écriture). Le vecteur (el_1, el_2, el_3) représente-t-il un état global du système qui a un sens ? Considérons le couple ordonné (el_2, el_1). D'une part tout message émis par P_2 vers P_1 dont l'émission est prise en compte dans el_2 a sa réception prise en compte dans el_1 et, d'autre part, tout message issu de P_2 dont la réception est prise en compte dans el_1 a son émission captée dans el_2. Du point de vue des communications de P_2 vers P_1, c'est-à-dire du canal c_{21}, nous dirons que el_2 est cohérent par rapport à el_1. De manière plus générale on introduit la notion de cohérence forte sur un couple ordonné d'états locaux (el_i, el_j) relatifs à P_i et P_j, m étant un message de calcul envoyé à P_j par P_i :

$$cohérent_F(el_i, el_j) \equiv \forall m \quad (\textit{émission(m) captée dans } el_i \Leftrightarrow \textit{réception(m) captée dans } el_j)$$

Considérons maintenant le couple ordonné (el_3, el_2) : el_3 rend compte de l'émission du message m_3, mais el_2 ne rend pas compte de la réception correspondante, on a donc $\neg cohérent_F(el_3, el_2)$; les deux états locaux " ne se correspondent pas " du point de vue des communications de P_3 vers P_2 ; une façon simple de rétablir la cohérence du couple ordonné (el_3, el_2) consiste à considérer que le message m_3 est en transit de P_3 vers P_2 (du point de vue des communications de P_3 vers P_2 et des états locaux el_3 et el_2). On dira que, par rapport au couple (el_3, el_2) le canal c_{32} (de P_3 vers P_2) contient le message m_3 ; de manière plus générale on définit l'état ec_{ij} d'un canal c_{ij} par rapport aux états locaux el_i et el_j comme la séquence des messages en transit relativement à ces états locaux (c'est-à-dire dont l'émission est captée dans l'état local el_i de l'émetteur et dont la réception ne l'est pas dans l'état local el_j du récepteur). Ceci conduit à la notion de cohérence faible d'un couple ordonné d'états locaux (m étant un message de calcul envoyé à P_j par P_i) :

$$cohérent_f(el_i, el_j, ec_{ij}) \equiv \forall m : (\textit{émission(m) captée dans } el_i \Leftrightarrow \textit{réception(m) captée dans } el_j \vee m \in ec_{ij})$$

on a bien sûr :

$$cohérent_F(el_i, el_j) \Rightarrow cohérent_f(el_i, el_j, ec_{ij})$$

Considérons maintenant le couple ordonné (el_2, el_3) : el_3 capte la réception d'un message (m_4) dont l'émission n'est pas prise en compte dans el_2 ; on a donc $\neg cohérent_f(el_2, el_3, c_{23})$. Le couple (el_2, el_3) est incohérent car il fait état d'un message reçu et non émis ; on ne peut comme précédemment considérer l'état du canal c_{23} pour rendre cohérent ce couple d'états locaux : m_4 ne peut être en transit puisque sa réception est prise en compte dans el_3 !

L'analyse qui précède apporte deux choses. La première est la définition d'un état global cohérent. Un état global EG est un ensemble composé d'un état local el_i pour chacun des sites P_i et d'un état ec_{ij} de chaque canal c_{ij} :

$$EG \equiv \left\{ \bigcup_{i \in [1,n]} el_i, \bigcup_{(i,j) \in \Gamma} ec_{ij} \right\}$$

La notion de cohérence généralement utilisée pour un état global peut être facilement définie à l'aide du prédicat $coh\acute{e}rent_f$. Un état global cohérent EG_f est tel que :

$$EG_f = \left\{ \bigcup_{i \in [1,n]} el_i, \bigcup_{(i,j) \in \Gamma} ec_{ij} \ / \ \forall (i,j) \in \Gamma : coh\acute{e}rent_f(el_i, el_j, ec_{ij}) \right\}$$

La cohérence d'un tel état global exprime d'une part que les états locaux mémorisés pour les sites ne font pas état de messages reçus et non émis et d'autre part que les messages émis et non reçus par rapport à ces états locaux définissent l'état des canaux relativement à cet état global.

Le second enseignement concerne les algorithmes distribués de calcul d'états globaux (cohérents). La mémorisation d'états locaux ne peut être faite de façon arbitraire ; afin que l'état global calculé soit cohérent il est nécessaire d'une part de capter l'état des canaux relativement à cet état global et d'autre part d'introduire des règles de synchronisation qui lient mémorisations d'états locaux et communications de messages, ceci afin d'assurer la cohérence (faible).

1.3 Algorithmes de calcul d'états globaux

Le premier algorithme de calcul d'un état global d'un système réparti a été proposé par Chandy et Lamport [2]. Cette solution suppose que les canaux de communication du système observé sont fiables, unidirectionnels et ne déséquencent pas les messages. Les règles de synchronisation introduites pour garantir la cohérence de l'état global capté utilisent des messages de contrôle (du type *marqueur*) et véhiculent ces messages sur les canaux de l'application. Les marqueurs effectuent un parcours structuré des canaux entre les sites, afin d'assurer d'une part que les messages émis et non reçus

(du point de vue des états locaux mémorisés) sont captés dans l'état des canaux correspondants, et d'autre part qu'il n'y a pas dans les états locaux mémorisés de messages reçus et non émis. Cette synchronisation utilise le fait que les canaux de l'application ne déséquencent pas les messages et que donc un message *marqueur* permet de distinguer les messages émis avant le marqueur de ceux émis après le marqueur. Un contrôleur placé sur chaque site peut capter son état local, celui de ses canaux entrants en fonction des marqueurs et de règles de synchronisation garantissant la cohérence de l'état global calculé.

Parmi les autres solutions proposées, l'algorithme de Lai et Yang [8] est particulièrement intéressant. Les canaux de communication, fiables et unidirectionnels, peuvent déséquencer les messages et l'algorithme n'utilise pas de messages de contrôle du type *marqueur*. Le contrôleur associé à chaque site doit stocker tous les messages émis (resp. reçus) sur chaque canal sortant (resp. entrant). Par rapport à la solution précédente celle-ci nécessite très peu de synchronisation mais le stockage des messages peut par contre la rendre impraticable. Les états globaux calculés par les deux algorithmes sont cohérents au sens de C_f.

1.4 Une classe remarquable d'états globaux

Quelle que soit l'utilisation que l'on désire faire des états globaux calculés il est important que ceux-ci soient les plus faciles à exploiter possible. Il n'est pas possible d'éliminer du calcul d'état global l'état des sites. Il est par contre possible de ne s'intéresser qu'à certains états globaux vérifiant une propriété, par exemple les états dans lesquels il n'y a pas de messages en transit (pour ces états les canaux sont vides).

De tels états globaux facilitent l'exploitation qui en est faite. Par exemple, la reprise après une panne ne requiert plus la restauration de l'état des canaux mais seulement la restauration de l'état des sites. La détection de la terminaison est simplifiée : tous les états locaux doivent indiquer la passivité des sites correspondants, aucun message en transit ne pouvant réactiver un site. Ces états globaux notés EG_F sont caractérisés par le prédicat $cohérent_F$:

$$EG_F \equiv \left\{ \bigcup_{i \in [1,n]} el_i / \forall (i,j) \in \Gamma : \ coh\acute{e}rent_F(el_i, el_j) \right\}$$

On s'intéresse dans cet article au calcul de tels états remarquables. On notera que toute exécution d'un système réparti passe par au moins deux tels états : son état initial et son état final. Il est possible que ces états soient les deux seuls états de ce type.

Le § 2 expose les principes algorithmiques à partir desquels il est possible de déterminer de tels états remarqables. La solution qui en découle nécessite de visiter tous les sites du système. Plusieurs algorithmes mettant en œuvre cette solution sont envisageables ; ils se distinguent par la structure de parcours choisie pour réaliser une telle visite (structure en anneau, structures arborescentes [7]). Nous examinerons au § 3 un de ces algorithmes utilisant un anneau de contrôle pour réaliser la visite des sites de l'application.

2 Principes algorithmiques

Afin de capter les états globaux dans lesquels il n'y a pas de messages de l'application en transit, on associe un contrôleur CTL_i à chaque site du système observé ; ce contrôleur a un rôle double.

Le premier rôle de CTL_i est un rôle d'observation locale ; il s'agit de capter l'état local el_i du site P_i. Comme nous l'avons indiqué, cet état local ne peut être capté arbitrairement ; il est nécessaire que la relation

$$P1(i) \equiv \bigwedge_{j \in X} coh\acute{e}rent_F(el_j, el_i)$$

soit vérifiée. Par convention, deux processus non voisins sont toujours mutuellement cohérents. En d'autres termes el_i doit rendre compte des réceptions de tous les messages que les autres sites P_j lui ont envoyés (et dont les émissions sont captées dans les el_j) et uniquement de celles-là. L'observation locale est donc assujettie à une contrainte globale. Pour mettre en œuvre ce contrôle on va utiliser les messages de contrôle que sont les marqueurs et les propriétés associées, de la façon suivante. L'émission de

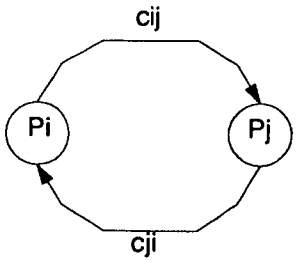

Figure 2 : Structure du système

marqueurs se fait lorsque un contrôleur CTL_j mémorise l'état local el_j du site P_j ; il émet alors un marqueur sur chacun des canaux de sortie de ce site (que nous désignerons par l'ensemble $c\_sortant_i$). Afin que la relation précédente soit vérifiée, la prise d'un état local el_i du site P_i par CTL_i n'est faite que lorsque CTL_i a reçu un marqueur sur chacun des canaux entrants du site P_i (désignés par $c\_entrant_i$). De plus un contrôleur CTL_i ne délivre au site P_i les messages arrivés après un marqueur qu'après avoir enregistré l'état el_i. Nous montrerons dans les preuves des algorithmes présentés (voir § 3.3) que cette synchronisation entre sites voisins, mise en œuvre par les contrôleurs et les marqueurs véhiculés sur les canaux du système observé, assure qu'un état local el_i vérifie la propriété $P1(i)$ dans laquelle el_j représente l'état local du site voisin P_j, capté juste avant l'envoi par CTL_j d'un marqueur sur le canal (P_j, P_i), dont la réception a permis à CTL_i de capter el_i.

Le second rôle des contrôleurs CTL_i a trait à une prise de décision globale. Considérons le vecteur V des derniers états locaux mémorisés par les sites :

$$V = (el_1, el_2, \ldots, el_j, \ldots el_i, \ldots, el_n)$$

Lorsque el_i a été mémorisé, CTL_j n'avait peut-être pas encore mémorisé el_j, le dernier état local mémorisé étant alors el'_j ; les règles de synchronisation précédentes assurent $cohérent_F(el'_j, el_i)$, mais qu'en est-il de $cohérent_F(el_j, el_i)$ (voir figures 2 et 3) ?

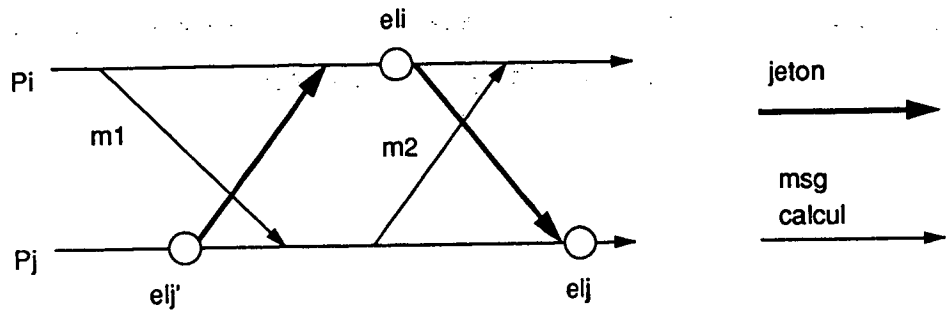

Figure 3 : Exemple d'exécution

On voit sur la figure 3 que le vecteur V des derniers états locaux mémorisés ne fait pas état de messages reçus et non émis, mais fait état de messages émis et non reçus, on a en effet $\forall(i,j) \in \Gamma$: $coh\acute{e}rent_f(el_j, el_i, c_{ji})$ (*cf.* § 3.3). En d'autres termes, bien qu'assujetties à des règles de synchronisation (utilisant des marqueurs), les mémorisations d'états locaux ne forment pas une opération globale atomique. Toutefois les règles précédentes assurent que dans tous les cas le vecteur V des derniers états locaux mémorisés $V = (el_1, el_2, \ldots, el_j, \ldots el_i, \ldots, el_n)$ ne fera jamais état de messages reçus et non émis.

Les contrôleurs doivent donc coopérer pour savoir si ce vecteur fait ou non état de messages émis et non reçus. Pour cela on introduit un contrôle supplémentaire : le comptage par chaque contrôleur CTL_i des nombres de messages émis et reçus par le site correspondant P_i. On définit les compteurs abstraits suivants :

$mt_{el_i}[j]$ = nombre de messages envoyés par P_i à P_j, via c_{ij}, relativement à el_i

$-mt_{el_i}[i]$ = nombre de messages reçus par P_i dans el_i, quel que soit l'émetteur

On a alors :

$$\sum_{\substack{el_k \in V \\ k \neq i}} mt_{el_k}[i] = \text{nombre de messages envoyés à } P_i \text{ par les } P_k \text{ relativement à leurs } el_k$$

$$-mt_{el_i}[i] = \text{nombre de messages reçus par } P_i \text{ dans } el_i$$

c'est-à-dire :

$$\sum_{el_k \in V} mt_{el_k}[i] = \text{nombre de messages en transit vers } P_i \text{ relativement au vecteur } V \text{ des états locaux mémorisés.}$$

Le vecteur V des derniers états locaux mémorisés représente alors un état global dans lequel il n'y a pas de messages émis et non reçus si et seulement si :

$$R \equiv \left(\forall i \in [1,n] : \sum_{el_k \in V} mt_{el_k}[i] = 0 \right)$$

Tous les contrôleurs sont mis en jeu lors du test de la valeur de cette relation R. La mise en œuvre de ce test nécessite donc de visiter tous les sites ; pour cela on peut utiliser des algorithmes distribués de parcours de réseau [13,12] ; ceux-ci peuvent s'appuyer sur une structure prédéfinie telle qu'un arbre de recouvrement ou un anneau construit sur le réseau. Nous examinons dans ce qui suit le cas où la visite est réalisée avec un anneau.

En résumé, les principes algorithmiques et les outils utilisés sont les suivants :

1. Une synchronisation entre sites voisins, fondée sur l'usage de marqueurs, qui d'une part définit les prises d'états locaux, et d'autre part garantit qu'il n'y a pas de messages reçus et non émis dans les états locaux mémorisés,

2. une coopération entre les sites, qui nécessite la gestion de compteurs de la part de chaque contrôleur, qui permet de savoir si il n'y a pas de messages émis et non reçus, relativement aux états locaux mémorisés ; si tel est le cas, ces états locaux sont alors sauvegardés.

Remarque 1. Si on ne considère que le point (1), on peut constater qu'il est possible d'obtenir un algorithme de calcul d'état global cohérent (comportant les états de sites et les états de canaux), dans lequel chaque site calcule l'état de ses canaux sortants relativement à EG_f. Cet algorithme est en quelque sorte le dual de celui de Chandy et Lamport dans lequel chaque site calcule l'état de ses canaux entrants.

Remarque 2. Le principe du comptage du nombre de messages dans les systèmes répartis a été utilisé par plusieurs auteurs ; citons notamment [4] et [9] en ce qui concerne la détection de la terminaison, et [5] qui aborde le problème plus général de la détection des propriétés stables.

3 Mise en œuvre avec un anneau

On présente ici un algorithme réparti de calcul d'états globaux, où il n'y a pas de messages de calcul en transit, dans lequel la valeur *vrai* ou *faux* de la relation R est calculée en visitant les sites à l'aide d'un jeton circulant sur un anneau virtuel construit sur le système observé[6]. Chaque contrôleur CTL_i est pourvu d'un tableau mt_i gérée comme indiqué au paragraphe précédent. Le jeton véhicule un vecteur *cmt* dans lequel sont accumulées les valeurs des vecteurs *mt* des sites qu'il visite. Si on associe une visite du jeton sur le site P_i à la mémorisation de l'état local el_i (c'est-à-dire en synchronisant la progression du jeton sur l'anneau et le balayage des canaux par les marqueurs), la valeur ajoutée à *cmt* représente alors mt_{el_i}.

3.1 Contexte des contrôleurs

Pour chaque site P_i l'application définit les canaux suivants :

$c\_entrants_i$ ensemble des canaux entrants dans P_i

$c\_sortants_i$ ensemble des canaux sortants de P_i

L'anneau virtuel utilisé par les contrôleurs est défini pour chaque site par :

c_pred_i élément de $c\_entrants_i$ utilisé par CTL_i pour recevoir le jeton

c_succ_i élément de $c\_sortants_i$ utilisé par CTL_i pour transmettre le jeton

3.2 L'algorithme

Le comportement d'un contrôleur CTL_i est alors le suivant :

1. De façon systématique : lorsque le processus P_i reçoit un message de calcul émis par P_j, le contrôleur CTL_i fait $mt_i[i] := mt_i[i] - 1$.

2. De façon systématique : lorsque le processus P_i émet un message de calcul vers un processus P_j, le contrôleur CTL_i fait $mt_i[j] := mt_i[j] + 1$.

3. Le texte principal exécuté par le contrôleur CTL_i est le suivant :

```
répéter
  attendre jeton(cmt) sur c_pred_i
  et attendre (marqueur) sur c: ∀c ∈ c_entrants_i;
  cmt := cmt + mt_i; — Opérations vectorielles
  mt_i := 0;
  <Mémoriser el_i>;
  si "tous les sites ont été visités une fois au moins par le jeton"
    et cmt = 0
    alors diffuser "Sauvegarder les el_i"
  fsi;
  envoyer jeton(cmt) sur c_succ_i et
  envoyer marqueur sur c : ∀c ∈ c_sortants_i
refaire;
```

Remarque sur l'initialisation. Le champ *cmt* du jeton et chaque vecteur mt_i sont initialisés avec un vecteur de zéros. L'initialisation de l'état des canaux est plus délicate : en effet, le premier contrôleur ayant le jeton attendra des marqueurs sur tous ses canaux entrants ; or l'émission de marqueurs n'est faite qu'au moment où un contrôleur envoie le jeton. Une solution simple consiste à restreindre l'ensemble des canaux entrants sur lesquels un contrôleur attend un marqueur en ne considérant que les sites déjà visités par le jeton une fois au moins. L'ensemble des sites visités une fois au moins par le jeton est facilement calculé en ajoutant un champ *visités* au jeton. Ce champ contient l'ensemble des noms des sites qui ont reçu le jeton une fois au moins. Il est initialisé avec la valeur *vide*. La ligne

> **attendre** *jeton(cmt)* **sur** c_pred_i
> **et attendre** *(marqueur)* **sur** $c : \forall c \in c\_entrants_i$

devient

> **attendre** *jeton(cmt,visités)* **sur** c_pred_i
> **et attendre** *(marqueur)* **sur** $c : \forall c \in c\_entrants_i \cap visités$;
> $visités := visités \cup \{i\}$;

et le test

> **si** "tous les sites ont été visités"

devient

> **si** $visités = X$

Remarque sur l'opération *diffuser*. Lorsqu'un contrôleur CTL_i constate que l'ensemble des états locaux mémorisés constitue un état global cohérent au sens de la cohérence forte, il exécute l'opération

> **diffuser** "Sauvegarder les el_i"

Cette opération peut être facilement mise en œuvre. Un contrôleur CTL_j doit recevoir un tel message de sauvegarde avant de mémoriser son prochain el_j ; le jeton de contrôle peut être mis à profit pour réaliser cette mise en œuvre.

3.3 Preuve

Il s'agit de prouver que l'algorithme assure les deux propriétés suivantes. Sur CTL_i on doit avoir :

$$P1(i) \equiv \bigwedge_{j \in X} cohérent_F(el_j, el_i)$$

Si $V = \{el_1, \ldots, el_n\}$ est un état global sauvegardé on doit avoir :

$$P2 \equiv \bigwedge_{(el_j, el_i) \in V} cohérent_F(el_j, el_i)$$

3.3.1 Preuve de *P1(i)*

Le raisonnement s'appuie sur un temps global abstrait, qui permet de formuler les relations de précédence des différents évènements du système. Des dates d'occurrence ne peuvent être comparées que lorsqu'elles traduisent une relation de causalité, c'est-à-dire se rapportent à des événements ayant lieu sur un même site ou bien sont des événements reliés par une transmission de message entre deux sites.

On pose :

$em_i(m_{ij})$ = date d'émission du message m_{ij} par P_i vers P_j
$rec_j(m_{ij})$ = date de réception du message m_{ij} par P_j

Propriétés dues aux hypothèses faites sur l'environnement (m_{ji}^1 et m_{ji}^2 étant deux messages quelconques émis par le site j vers le site i) :

Propriété 3.1 *Les canaux ne déséquencent pas les messages.*

$$em_j(m_{ji}^1) < em_j(m_{ji}^2) \Leftrightarrow rec_i(m_{ji}^1) < rec_i(m_{ji}^2) \quad (1)$$

D'après le texte de l'algorithme :

Propriété 3.2 *Un marqueur mq_{ji} reçu par CTL_i est toujours reçu avant l'instant del_i où CTL_i prend l'état local el_i de P_i.*

$$rec_i(mq_{ji}) < del_i \quad (2)$$

Toujours d'après le texte de l'algorithme :

Propriété 3.3 *Un marqueur mq_{ji} émis par CTL_j est toujours émis après l'instant del_j où CTL_j prend l'état local el_j de P_j :*

$$em_j(mq_{ji}) > del_j \tag{3}$$

D'après sa définition $coh\acute{e}rent_F(el_j, el_i)$ exprime (mc désignant un message de l'application) :

$$em_j(mc_{ji}) < del_j \Rightarrow rec_i(mc_{ji}) < del_i \tag{4}$$
$$rec_i(mc_{ji}) < del_i \Rightarrow em_j(mc_{ji}) < del_j \tag{5}$$

Nous allons prouver $P1(i)$ en prouvant que les relations 4 et 5 sont vérifiées. Prouvons 4 par contradiction : on suppose

$$em_j(mc_{ji}) < del_j \wedge rec_i(mc_{ji}) > del_i' \tag{6}$$

La relation 6 et la propriété 3.2 impliquent

$$em_j(mc_{ji}) < del_j \wedge rec_i(mc_{ji}) > rec_i(mq_{ji}) \tag{7}$$

La relation 7 et la propriété 3.3 impliquent

$$em_j(mc_{ji}) < em_j(mq_{ji}) \wedge rec_i(mc_{ji}) > rec_i(mq_{ji}) \tag{8}$$

Or cette relation est en contradiction avec la propriété 3.1 et est donc fausse. Sa négation est vraie et ce n'est autre que 4

$$em_j(mc_{ji}) < del_j \Rightarrow rec_i(mc_{ji}) < del_i \tag{9}$$

Prouvons maintenant 5 par contradiction. On suppose vraie la relation suivante :

$$rec_i(mc_{ji}) < del_i \wedge em_j(mc_{ji}) > del_j \tag{10}$$

Or d'après l'algorithme on a la propriété (en raison de l'atomicité du traitement effectué lors de la réception du jeton)

$$em_j(mc_{ji}) > del_j \Rightarrow em_j(mc_{ji}) > em_j(mq_{ji}) \tag{11}$$

La relation 10 et la propriété 11 impliquent

$$rec_i(mc_{ji}) < del_i \wedge em_j(mc_{ji}) > em_j(mq_{ji}) \tag{12}$$

L'atomicité de l'exécution assure

$$rec_i(mc_{ji}) < del_i \Rightarrow rec_i(mc_{ji}) < rec_i(mq_{ji}) \tag{13}$$

La relation 12 et la propriété 13 impliquent

$$rec_i(mc_{ji}) < rec_i(mq_{ji}) \wedge em_j(mc_{ji}) > em_j(mq_{ji}) \tag{14}$$

Cette dernière relation est en contradiction avec la propriété 3.1. La relation 10 est donc fausse, sa négation est vraie, ce n'est autre que 5.

3.3.2 Preuve de $P2$

La sauvegarde de l'état global est décidée par un contrôleur CTL_i au moment où il constate que le vecteur cmt est nul ; on notera cmt_{el_i} la valeur de cmt à cet instant. Prouver $P2$ revient à prouver l'équivalence

$$cmt = 0 \Leftrightarrow \forall (j,k) \in X^2 : \ coh\acute{e}rent_F(el_j, el_k) \tag{15}$$

Pour simplifier la notation on suppose dans la suite que sur l'anneau le prédécesseur d'un site P_i est P_{i-1} et que son successeur est P_{i+1} (modulo n). Prouvons tout d'abord

$$\forall (j,k) \in X^2 : \ coh\acute{e}rent_F(el_j, el_k) \Rightarrow cmt = 0 \tag{16}$$

La propriété $P1(i)$ étant vraie pour tout i, il n'y a aucun message en transit relativement au vecteur V des el_i, et ceci implique $cmt = 0$. Prouvons la réciproque de 16 :

$$cmt = 0 \Rightarrow \forall (j,k) \in X^2 : \ coh\acute{e}rent_F(el_j, el_k) \tag{17}$$

On suppose que le contrôleur qui détecte $cmt = 0$ est CTL_i. Désignons par cmt_{el_i} la valeur de cmt au moment où CTL_i effectue le test sur cmt. On raisonne par contradiction : supposons vraie la relation pour un j donné

$$cmt_{el_i}[j] = 0 \wedge \exists k : \neg coh\acute{e}rent_F(el_k, el_j) \tag{18}$$

Deux cas se présentent sur l'anneau en fonction de la postion de i, j et k :

1. cas $j < k < i$
   On a nécessairement $cmt_{el_j}[j] = 0$ lorsque le jeton quitte CTL_j (on a alors $P1(j)$) et seul CTL_j peut faire décroître $cmt[j]$. On a aussi

$$\exists k : \neg cohérent_F(el_k, el_j) \Rightarrow cmt_{el_k}[j] > 0 \qquad (19)$$

   Lors de son trajet entre P_k et P_i la valeur de $cmt[j]$ ne peut pas décroître et donc la relation 18 est fausse pour le cas $j < k < i$.

2. cas $k < j < i$
   Les messages envoyés par P_k à P_j sont captés dans les états locaux (l'émission est captée dans el_k et la réception dans el_j). On a donc $cohérent_F(el_k, el_j)$. Dans ce cas la relation 18 est fausse également pour les k tels que $k < j < i$.

   Les cas où i est inférieur à j ou à k ne sont pas à considérer car les messages en transit qui sont la condition nécessaire de $\neg cohérent_F(el_k, el_j)$ sont partis après el_k et donc ne concernent pas ce tour du jeton.

Le raisonnement qui vient d'être fait vaut pour tout j :

$$\forall j : (cmt_{el_i}[j] = 0 \Rightarrow \forall k : cohérent_F(el_k, el_j)) \qquad (20)$$
$$\equiv (cmt_{el_i} = 0 \Rightarrow \forall (j,k) : cohérent_F(el_k, el_j)) \qquad (21)$$

Q.E.D.

4 Conclusion

Le problème de la détermination d'états globaux cohérents est fondamental dans le contexte des applications et des systèmes répartis. Après avoir caractérisé de tels états nous nous sommes intéressés à la détermination d'une classe particulière d'états globaux cohérents : ceux dans lesquels il n'y a pas de messages en transit. Ces états globaux sont particulièrement intéressants ; ils permettent par exemple de définir des points de contrôle ou de reprise et facilitent la détection des propriétés stables. Il est important de remarquer que la propriété qui caractérise ces états (*i.e.* les canaux sont vides) n'est pas une propriété de stabilité.

L'algorithme proposé pour capter de tels états s'appuie sur une démarche méthodique :

- Le problème a tout d'abord été caractérisé par les propriétés que doivent respecter d'une part la prise d'un état local (propriété $P1(i)$) et d'autre part l'ensemble des états locaux mémorisés (propriété $P2$).

- La propriété $P1(i)$ définit la prise cohérente d'un état local du site P_i. Garantir $P1(i)$ repose, comme nous l'avons vu, sur l'algorithmique des messages de contrôle que sont les marqueurs, véhiculés sur les canaux, sans déséquencement de message, utilisés par l'application. (Cette technique a été initialement proposée dans [10] et développée dans [2] et [3].)

- Le test de la propriété $P2$ nécessitant un parcours de tous les sites, plusieurs mises en œuvre sont possibles en fonction du parcours choisi ; l'algorithme utilisant un parcours sur un anneau a été présenté et prouvé, d'autres algorithmes fondés sur les mêmes propriétés et utilisant d'autres parcours (arbre) sont envisageables. Les propriétés $P1(i)$ et $P2$ définissent en fait une classe d'algorithmes.

Outre le calcul d'état globaux remarquables, la famille d'algorithmes proposée est intéressante car elle offre un solution simple au problème des captures répétées d'états globaux : l'algorithme présenté dans [2] ne permet en effet de capter qu'un seul état global (à notre connaissance, la seule solution proposée pour résoudre ce problème figure dans [1] ; elle fait l'hypothèse de la communication par rendez-vous, ce qui élimine le problème posé par l'état des canaux ; la classe d'algorithmes proposée est donc plus générale que cette solution pour résoudre le problème des captures répétées).

Bibliographie

[1] L. Bougé. Repeated snapshots in distributed systems with synchronous communications and their implementation in CSP. *Theor. Computer Science*, 49:145–169, 1987.

[2] K. M. Chandy and L. Lamport. Distributed snapshots: determining global states of distributed systems. *ACM TOCS*, 63–75, February 1985.

[3] K. M. Chandy and J. Misra. An example of stepwise refinement of distributed programs: quiescence detection. *ACM TOPLAS*, 8(3), July 1986.

[4] E. W. D. Dijkstra and C. S. Scholten. Termination detection for diffusing computation. *Information Processing Letters*, 11:217–219, August 1980.

[5] J.M. Helary, C. Jard, N. Plouzeau, and M. Raynal. Detection of stable properties in distributed applications. 6^{th} *ACM SIGACT-SIGOPS, Symp. Principles of Distributed Computing, Vancouver, Canada*, August 1987.

[6] J.-M. Hélary and M. Raynal. *Depth-first Traversal and Virtual Ring Construction in Distributed Systems.* Rapport de recherche 369, IRISA, July 1987.

[7] J.-M. Helary and M. Raynal. *Synchronisation et contrôle des systèmes et des programmes répartis.* Eyrolles, 1988. à paraître, 200 p.

[8] T.H. Lai and T.H Yang. On distributed snapshots. *Inf. Proc. Letters*, 25:153–158, 1987.

[9] F. Mattern. Experience with a new distributed termination detection algorithm. In *Proc. of* 2^{nd} *Int. Workshop on Distributed Algorithms, Amsterdam*, 1987.

[10] J. Misra. Detecting termination of distributed computations using markers. In *Proc. ACM Symposium on PODC, Montréal*, pages 290–294, August 1983.

[11] C. Morgan. Global and logical time in distributed algorithms. *Inf. Proc. Letters*, 20:290–294, 1985.

[12] M. Raynal. *Systèmes répartis et réseaux : concepts, outils et algorithmes.* Eyrolles, Février 1987.

[13] A. Segall. Distributed networks protocols. *IEEE Transactions on Inf. Theory*, IT29(1):23–35, January 1983.

[14] G. Tel. *Distributed Infimum Approximation.* Tech. report RUU-CS-86-12, University of Utrecht, 1986.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique